トランザクションの一貫性となる基準時点

【トランザクションの分離レベル】
Select 操作での読取られるレコードの基準時点の指定

SELECT 操作のレコードの基準時点（時系列データの一貫性確保）

分離レベル名	内　　容
コミット読取り（デフォルト）	SELECT 発行を行った時点に、Oracle システムの現時点の SCN 番号を入手する この SELECT 文の読み込み処理が、対象データが大量にあり長時間かかる場合には、入手している SCN 番号以下の値の COMMIT 済のデータを読み取る（SCN 値による時系列データの一貫性確保） ※1 同一 SELECT 文を２回発行させた場合に、その SELECT 文の実行の間に、他者がデータを更新、挿入、削除して COMMIT した場合、後の SELECT の実行結果には、COMMIT 後の最新のデータが読み込まれてくる
シリアライズ可能	最初に SET　TRANSACTION を使って、Oracle システムの現時点の SCN 番号を入手する これ以降の SELECT 文の処理は、すべて入手している SCN 番号以下の値の COMMIT 済のデータを読み取る SELECT 文が複数存在し順次発行していく処理の場合、使用する SCN 番号は、SET TRANSACTION で入手したときの同一の SCN 番号が使われていく なので、上記※1 のような現象は発生しない ※2 他者が先に更新してしまったレコード（commit が未、済）に対して、削除、更新処理を行う操作を発行した場合には、このトランザクションの処理に対しエラー割込みが発生する（ORA-08177）
読取り専用	最初に SET　TRANSACTION を使って、Oracle システムの現時点の SCN 番号を入手する これ以降の SELECT 文の処理は、すべて入手している SCN 番号以下の値の COMMIT 済のデータを読み取る SELECT 文が複数存在し順次発行していく処理の場合、使用する

	SCN 番号は、SET TRANSACTION で入手したときの同一の SCN 番号が使われていく 　なので、上記※1 のような現象は発生しない 　なお、「読取り専用」分離レベルを採用しているトランザクションでは、自身のトランザクション内で DELETE、UPDATE、INSERT 処理は出来ない。 「シリアライズ可能」よりパフォーマンスが優れる 　上記※1、2 のような現象は、発生しない ~~また、他に並行して動作しているトランザクションに対しても、「読取り専用」分離レベルのトランザクションが終了するまでは、処理開始時のデータを、SELECT 発行時に提供させる~~

SELECT で使用される SCN 番号

システム 最新 SCN	他のユーザー の処理	デフォルトでの SELECT 命令の動き	SET TRANSACTION を使った時の SELECT
1			
1			SET TRANSACTION SCN 番号＝1 を取得
1		Select 命令発行 SCN 番号＝1 を取得 SCN 番号＝１以下の値のレコードの読み込み ↓	
1	INSERT 処理 SCN 番号が 2 でセット		
②	COMMIT 発行		Select 命令発行 SCN 番号＝１以下の値のレコードの読み込み ↓
2			
2	UPDATE 処理 SCN 番号が 3 でセット		
2		Select 命令発行 SCN 番号＝2 を取得 SCN 番号＝2 以下の値のレコードの読み込み ↓	
2			Select 命令発行 SCN 番号＝１以下の値のレコードの読み込み ↓
③	COMMIT 発行		

トランザクションへの分離レベルの指定方法（時系列データの一貫性確保）

・「コミット読取り」分離レベル
　　自動開始　（デフォルト）

・「シリアライズ可能」分離レベル
　　SET　TRANSACTION　ISOLATION　LEVEL　SERIALIZABLE；

・「読取り専用」分離レベル
　　SET　TRANSACTION　READ　ONLY；
　　※ 処理終了後、COMMIT 操作を行ってトランザクションが行っている読取り一貫性の制御を外す必要がある

【注意】

コネクションプール（共有サーバー接続）を使用した環境では、明示的に~~「シリアライズ可能」~~、「読取り専用」分離レベルを完了しなかった場合、別のトランザクションがコネクションプール（共有サーバー接続）を再利用した時に「読取り専用」分離レベルが引き継がれてしまい、予期せぬ不具合が発生します。

発生を防ぐためには、処理を終了させる時に COMMIT 命令を発行する

UNDO 表領域とは

Select 文の処理において、処理が複雑もしくは大量データにより長時間の実行となるものについては、以下の動作を行い、取得されるデータの同一時刻での一貫性を保証させます

1）Select 文のオラクル内部処理では、Select 文を開始した時点に、『最初のページに記述した仕組み』で Oracle システムの SCN 番号を入手する

2）他のセッションの処理でデータが更新された場合、Oracle システムでは更新データと SCN 番号がセットで管理されていきます

3）Select 文でデータの読取りを行う時に、更新処理が **Select で使用する SCN 番号より大きい**場合には、UNDO 表領域を使って更新前データに戻して、Select 文の処理結果としてサーバープロセスに渡していきます

UNDO 表領域は、表のデータであるレコードの**更新前の値**が保存されているシステム専用のエリアです（時系列データの一貫性確保）

エラー・コードとその原因

ORA-00054:　リソースビジー、NOWAIT が指定されていました

Select　・・・・　For　Update　NOWAIT コマンドで読み込もうとしたレコードは、排他ロックが掛っており読込みが出来なかった

ORA-03006:　リソースビジー、WAIT タイムアウトの期限に達しました

Select　・・・・　For　Update　WAIT　秒数 コマンドで読み込もうとしたレコードは、指定秒数を待っても排他ロックが解除されず読込みが出来なかった

ORA-08177:　このトランザクションのアクセスをシリアル化できません

前ページ 参照のこと

分離レベル名：シリアライズ可能において、他者が先に更新してしまったレコード（commit 未 and 済）に対して、削除、更新処理を行おうとした

ORA-01456:　READ ONLY トランザクションでは挿入/削除/更新ができません。

分離レベル名：読取り専用において、挿入/削除/更新操作を行おうとした

ORA-01555 スナップショットが古すぎます

これは、UNDO 表領域の再利用が行われ、更新前データが上書きされ残っていないので、「(トランザクション開始時点からの）読取り一貫性が出来なかった」ということです

この場合、エラーが発生した SELECT 処理が、異常終了となります。

※ エラーが発生した処理において、このセッションのエラー発生前に行った更新等の処理（UPDATE など）については、アプリケーションのエラー・ハンドリング処理を記述し、COMMIT させるか ROLLBACK させるかを決定し指定させる必要があります

ORA-30036 UNDO 表領域が拡張できません

これは、更新系の処理において起こるエラーです

トランザクションでデータの更新をしようとしたときに、更新前の状態を UNDO 表領域に保存します

このときに、UNDO 表領域が一杯だったので表領域拡張を行おうとして領域確保が出来なかったエラーです

UNDO 領域が解放されるまでの更新からの経過時間

UNDO 領域が解放されるまでの更新からの経過時間

初期化パラメータ undo_retention で、保持する時間（単位：秒）を指定します
ただし、これは目安時間なので、UNDO 領域が不足した場合には、上書きされます

初期化パラメータ undo_retention の設定されている値の確認

```
show　parameter　undo_retention
```

更新前データを確保していた実際の有効時間数の確認

これは、設定値の時間数の更新前データが確実に保証されているのではなく、実際に
確保されている時間数は更新データのデータ量の多少により変化していく
この実際の更新前データ確保の時間保証数の時間経過変動を確認する

```
alter　session　set　NLS_DATE_FORMAT　=　'yyyy/mm/dd hh24:mi:ss' ;

select　BEGIN_TIME ,　END_TIME ,　TUNED_UNDORETENTION
　　from　V$UNDOSTAT
　　order by　BEGIN_TIME　desc ;

　　--　or　--

select　BEGIN_TIME ,　END_TIME ,　TUNED_UNDORETENTION
　　from　DBA_HIST_UNDOSTAT
　　order by　BEGIN_TIME　desc ;
```

BEGIN_TIME	END_TIME	TUNED_UNDORETENTION
2018/08/02 10:47:56	2018/08/02 10:52:22	1991
2018/08/02 10:37:56	2018/08/02 10:47:56	1750
2018/08/02 10:27:56	2018/08/02 10:37:56	1150
2018/08/02 10:17:56	2018/08/02 10:27:56	1725

エラーが発生しないための対応方法　　３種

（A）UNDO 表領域の拡張
（B）UNDO 表領域の自動拡張の採用
（C）初期化パラメータ undo_retention で指定した**保持時間内の絶対確保**
UNDO 表領域に、RETENTION_GUARANTEE を設定する
※ 参照処理が優先の意味になる

保持時間内の絶対確保を設定した時の弊害

UNDO 表領域の容量は決まっている
よって、UNDO 表領域に上書きする領域が無いという現象が発生する
すなわち、UPDATE 処理が、更新前データを保持する領域が確保できない別の障害が、発生する

【SQL 文の分割によりアクセスブロックを減少させる方法】

~~【Select 対象になったレコードの更新禁止設定】~~
【読み取り一貫性機能による対象処理データの時刻タイミングの固定】

複雑なロジックが必要な統計値などの値を複数求める場合、一括して１つの SQL 文に複数の関数を使って同時に計算させと、全件検索によりアクセス効率が低下する。

対応としては、SQL 文を分割する
すなわち、求めたい値の１つずつに単独の Select 文を発行する。

この時に、次の SQL 文を発行する間に、途中でレコード追加や変更が起きると、調査データの母体が異なってしまい、統計値が不整合な結果となってしまう。
このようなことが無いように、発行する SQL 文が同じ調査データの母体を保てるように最初に、「 SET　TRANSACTION　READ　ONLY 文 」を発行して、Oracle の読取り一貫性制御をつかって、データの時刻タイミングを固定して、処理の一貫性を確保する必要がある

使用方法）

```
SET　TRANSACTION　READ　ONLY　;

SELECT 文　1
SELECT 文　2
SELECT 文　3

COMMIT　;
```

例えば、化学と歴史の各教科の最高点と最低点を求める場合に、
下記の１つの Select 文よりも４つに分割した SQL 文の方が、アクセス効率がよい

```
Select　MAX( sience )，　MIN( sience )，　MAX( history )，　MIN( history )
    from　TEST_DATA；
              ↓
Select　MAX( sience )　from　TEST_DATA；
Select　MIN( sience )　from　TEST_DATA；
Select　MAX( history )　from　TEST_DATA；
Select　MIN( history )　from　TEST_DATA；
```

　このときに重要になるのが、1つめのSQL文を処理した後でかつ最後のSQL文の処理をしている最中に対象データの更新が発生して、前提レコード条件が異なってしまうことである
　このために、検索対象となる**時刻タイミングを固定**して、更新されたデータについては時刻固定された時点のデータで処理を行わせる

SET　TRANSACTION　READ　ONLY文は、対象データの更新を禁止しているわけではない。

以降で行われるSelect文の処理が**同一状態（同一時点）のデータを対象**にさせるようにするために、更新されたデータの更新前状態のデータを使用するようにと指定している